

MicroTCA AMC523 Bare DDCP

Hardware User Guide



Fermilab, United States Department of Energy

Ryan Rivera

Jonathan Eisch

beams-doc-####

Created: 24 August 2020

Last Update: 24 August 2020

Document Change Log

DocDB Version	Date	Pages	Editor	Description of Changes
V1	24 Aug 2020	All	rrivera	Created.

1. Introduction

The AMC523 Bare DDCP module provides an interface between the AMC523 Advanced Mezzanine Card (with MRT523b MTCA.4 Rear Transition Module) and DDCP clients. The AMC523 and MRT523b are both VadaTech cards; more information on the MicroTCA evaluation and demonstration effort can be found at beams-doc-7977.

The key features of the AMC523, as related to the MicroTCA evaluation and demonstration effort, include the following:

- Two 1000Base-X gigabit Ethernet ports on AMC ports 0 and 1.
- DDR3 2GB of memory.

The key features of the MRT523b, as related to the MicroTCA evaluation and demonstration effort, include the following:

- 12-channel (3 chips) of ADC with 16-bit samples at 125MHz.

2. Functional Description

The AMC523 Bare DDCP module firmware is based on the VadaTech amc523_xxx_42x_xxx_mrt523b Reference Design targeted for the Xilinx Kintex-7 FPGA xc7k410tffg900-2. For the Bare DDCP functionality project, all features except for the 1000Base-X gigabit Ethernet port on AMC port 0 were commented out. The pure RTL otsdaq ethernet controller was inserted, and a pure RTL DDCP data manager was developed to provide the DDCP network layer to act as a DDCP server. The otsdaq ethernet controller interfaces to the 1000Base-X Xilinx IP block using GMII.

The DDCP server functionality is split between two virtual slots: 0 and 1.

Slot 0 is the user firmware address space and for the Bare DDCP project will return 64-bit quadwords for every read that mirror the target feature and index. Slot 0 writes can be used to trigger a DDCP interrupt.

Slot 1 is the ethernet control address space. Here the otsdaq ethernet controller features can be accessed, as well as the DDCP interrupt destination.

3. External Connections

The AMC523 module is equipped with a custom MTCA.4 RTM card, the MRT523b. For the Bare DDCP project, the MRT523b is ignored.

4. Operation

ARP Operation

The otsdaq ethernet controller handles ARP replies and gratuitous ARPs on reset or on address change. Network hardware will automatically make ARP requests to locate the MAC address of a target IP address.

ICMP Ping Operation

The otsdaq ethernet controller handles ICMP ping replies completely internally.

For example, to ping the AMC with an IP address of 10.0.0.15 use the following command:

```
ping 10.0.0.15
```

A successful response will look like this:

```
PING 10.0.0.15 (10.0.0.15) 56(84) bytes of data.  
64 bytes from 10.0.0.15: icmp_seq=1 ttl=128 time=0.104 ms  
64 bytes from 10.0.0.15: icmp_seq=2 ttl=128 time=0.063 ms
```

DDCP Read Operation

From the DDCP client point of view reads are in units of 64-bit quadwords. There are two virtual slots that can be read: 0 and 1. Both virtual slots have a 32-bit address space formed with an 8-bit feature and 24-bit index.

Virtual Slot 0 reads should expect the feature and index mirrored back in the data.

For example, using the ddcprw tool one can read (option -r) a count of 4 quadwords (option -c4 and -d64) from an AMC at 10.0.0.15 (option -@10.0.0.15), virtual slot 0 (options -s0), starting at feature 1 (option -f1) and index 8 (option -i8), by using the following command:

```
ddcprw -@10.0.0.15 -s0 -f1 -i8 -c4 -d64 -r
```

A successful response will look like this:

```
Updated DDCP environment...  
  server: 10.0.0.15, MTU: 9000, slot: 0, feature: 1, index: 8, count: 4,  
width: d64  
Read 4 feature #1 d64 location(s)  
Note: indices are displayed in decimal, data are displayed in hexadecimal.  
   8: 0000000001000008 0000000001000009  *.....*  
  10: 000000000100000a 000000000100000b  *.....*
```

Virtual Slot 1 reads should expect this address space definition:

<i>Index</i>	<i>Description</i>
0	Upper 24-bits of self IP Address
1	Lower 7-bits of self IP and MAC Address
2	Upper 40-bits of self MAC Address
100 (0x64)	DDCP Interface Version (Read-only)

DDCP Query Operation

From the DDCP client point of view queries are defined for any defined read address. The query response will always indicate the supported operations are Query, Read, and Set.

For example, using the ddcprw tool one can query (option -q) an AMC at 10.0.0.15 (option -@10.0.0.15), for supported operations of virtual slot 0 (option -s0), feature 1 (option -f1), and index 8 (option -i8), by using the following command:

```
ddcprw -@10.0.0.15 -s0 -f1 -i8 -c1 -d64 -q
```

A successful response will look like this:

```
Updated DDCP environment...
  server: 10.0.0.15, MTU: 9000, slot: 0, feature: 1, index: 8, count: 1,
width: d64
Query node 10.0.0.15, slot #0, feature #1...
supportedOps: Query Read Set, elementBytes: 8, elements: 1
```

DDCP Write Operation

From the DDCP client point of view writes are in units of 64-bit quadwords. There are two virtual slots that can be written: 0 and 1. Both virtual slots have a 32-bit address space formed with an 8-bit feature and 24-bit index.

Virtual Slot 0 writes will trigger an interrupt corresponding to the least significant 4-bits of data, regardless of feature or index. Available interrupts are 0-31.

Virtual Slot 1 writes should expect this address space definition:

<i>Index</i>	<i>Description</i>
0	Upper 24-bits of self IP Address
1	Lower 7-bits of self IP and MAC Address
2	Upper 40-bits of self MAC Address
13 (0xD)	DDCP Interrupt Destination Capture and Interrupt Enable Mask

For example, using the ddcprw tool one can write/poke (option -p) a count of 2 quadwords (option -c2 and -d64), with value 0x400 and 0xA00, to an AMC at 10.0.0.15 (option -@10.0.0.15), virtual slot 0 (options -s0), starting at feature 1 (option -f1) and index 8 (option -i8), by using the following command:

```
ddcprw -@10.0.0.15 -s0 -f1 -i8 -c2 -d64 -p 0x400 0xA00
```

A successful response will look like this:

```
Updated DDCP environment...
  server: 10.0.0.15, MTU: 9000, slot: 0, feature: 1, index: 8, count: 2,
width: d64

Patch 2 feature #1 d64 location(s)
User provided values: 0000000000000400 0000000000000a00
```

DDCP Interrupt Operation

DDCP interrupts are sent from the AMC523 module as though the AMC523 is a DDCP client to a predetermined DDCP server (i.e. a UDP listener on port 65000). Note that all other DDCP operations (i.e. Read, Query, and Write) are usually conducted from the perspective that the AMC523 is a DDCP server and the interacting UDP process is the DDCP client.

For an interrupt to be successful, the interrupt destination must be setup in advance. This is done by writing a quadword to virtual slot 1, feature 0, index 13 (0xD). Note that the interrupt destination MAC and IP address are automatically captured during this write operation (i.e. they are not specified in the quadword bitfield), and the DDCP interrupt destination port is always 65000 (since the destination is considered a DDCP server). The bit definition of the quadword are as follows:

Bits	Description of virtual slot 1, feature 0, index 13 (0xD) bitfields
63:32	32-bit disable mask for interrupts 0-31 (0 is enabled, 1 is disabled)
31:16	16-bit interrupt destination slot
15:0	16-bit interrupt destination feature

Following the setup of the interrupt destination, any writes to virtual slot 0 will trigger an interrupt. The interrupt number 0-31 is specified by the least significant 4-bits of the data written. Note that feature and index are ignored.

For example, using the ddcprw tool one can force interrupt #4 from an AMC at 10.0.0.15 (option -@10.0.0.15), to be sent to the commanding host's feature 0 and slot 0, by using the following commands:

```
ddcprw -@10.0.0.15 -s1 -f0 -i13 -c1 -d64 -p 0x0 #set interrupt destination
ddcprw -@10.0.0.15 -s0 -f0 -i0 -c1 -d64 -p 0x4 #trigger interrupt 4
```

5. Firmware Programming

The AMC523 module uses the front-panel JTAG connector for programming bit-files to the FPGA, and for accessing the internal logic analyzer. At this time the Flash PROM has not been explored and still has the VadaTech supplied example bitstream programmed.